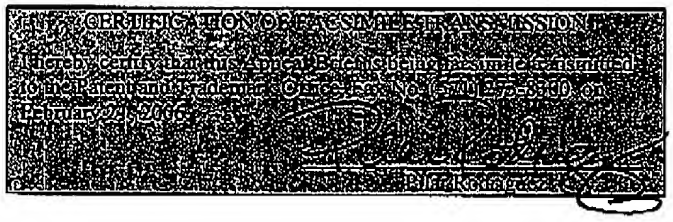
Patent

Attorney Docket No.: Intel 2207/7083
Serial No.: 09/750,095
Assignee: Intel Corporation

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT	:	STEPHAN J. JOURDAN et al.
SERIAL NO.	:	09/750,095
FILING DATE	:	December 29, 2000
GROUP ART UNIT	:	2183
FOR	:	METHOD AND APPARATUS FOR A REGISTER RENAMING STRUCTURE
EXAMINER	:	Richard L. Ellis

M/S: APPEAL BRIEFS - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450



APPEAL BRIEF

Dear Sir:

This brief is in furtherance of the Notice of Appeal, filed in this case on November 18, 2005.

1. REAL PARTY IN INTEREST

The real party in interest in this matter is Intel Corporation. (Assignment recorded April 23, 2001, Reel/Frame 011787/0521).

02/23/2006 EFLORES 00000145 110600 09750095
01 FC:1402 500.00 DA

2. RELATED APPEALS AND INTERFERENCES

There are no related appeals.

3. STATUS OF THE CLAIMS

Claims 1-5, 7-14 and 16-22 are pending in this application. Claims 1-5, 7-12 and 19-22 are allowed. Claims 13, 14 and 16-18 are rejected.

4. STATUS OF AMENDMENTS

The claims listed on page 1 of the Appendix attached to this Appeal Brief reflect the present status of the claims.

5. SUMMARY OF THE CLAIMED SUBJECT MATTER

The embodiment of claim 1 of the present application generally describes a processor, comprising a physical register file populated by a number of registers (e.g., *see* page 6, lines 14-17 – Figure 3, 212); an instruction decoder (e.g., *see* page 6, lines 11-12); a register alias table coupled to the instruction decoder (e.g., *see* page 6, lines 14-15); an active list populated by a number of entries (e.g., *see* page 6, lines 27-28 – Figure 3, 216), the entries include an old field and a new field (e.g., *see* page 6, lines 28 – Figure 3, 218, 220); a free list of unallocated physical registers reclaimed from said active list (e.g., *see* page 6, lines 28-29 – Figure 3, 214); and a misprediction condition wherein said free list reclaims mispredicted allocated said physical registers from said new field (e.g., *see* page 7, lines 8-9).

The embodiment of claim 8 of the present application generally describes a method for recovering registers in a processor, comprising detecting an exception correlating to an

instruction associated with an entry on an active list (e.g., *see* page 6, lines 28-29 – Figure 3, 214); moving a pointer on said active list to an old field and a new field after said entry (e.g., *see* page 6, lines 28 – Figure 3, 218, 220); and reclaiming during a misprediction condition allocated physical registers in said new field to a free list (e.g., *see* page 7, lines 8-9).

The embodiment of claim 13 of the present application generally describes a method for recovering registers in a processor, comprising reading a bit in an active list (e.g., *see* page 6, lines 28-29 – Figure 3, 214); reclaiming a physical register from a new field of said active list to a free list according to said bit (e.g., *see* page 7, lines 8-9); and setting said bit during a misprediction condition (e.g., *see* page 7, lines 8-9).

The embodiment of claim 19 of the present application generally describes a register renaming apparatus within a processor, comprising a register alias table (e.g., *see* page 6, lines 14-15); a first set of registers to be renamed by said register alias table(e.g., *see* page 6, lines 10-12); an active list having (e.g., *see* page 6, lines 27-28 – Figure 3, 216) an old field and a new field that correlate to said registers (e.g., *see* page 6, lines 28 – Figure 3, 218, 220); and a free list comprising a second set of registers reclaimed from said active list (e.g., *see* page 6, lines 28-29 – Figure 3, 214); and a misprediction condition wherein said free list is to reclaim mispredicted allocated said registers from said new field (e.g., *see* page 7, lines 8-9).

Fig. 3 is a block diagram of a processor 200 in accordance with an embodiment of the present invention. Processor 200 may be an out-of-order processor. For simplicity, not all functional units of processor 200 are depicted. In processor 200, fetch unit 202 fetches instructions from instruction cache 204, and decode unit 206 decodes these instructions. For a Complex Instruction Set Computer ("CISC") architecture, decode unit 206 decodes a complex

instruction into one or more micro-instructions. The micro-instructions may define a load-store type architecture, so that micro-instructions involving memory operations are simple load or store instructions. The present invention, however, may be practiced for other architectures, such as Reduced Instruction Set Computer ("RISC"), Very Large Instruction Word ("VLIW") architectures, and the like.

A register renaming functional unit 208 may include register allocation table ("RAT") 210. RAT 210 contains current mappings between logical registers and physical registers. The physical registers are indicated by physical register file 212. The decoded instructions may indicate the logical, or architectural, registers to be utilized by the instruction. Every logical register should have a mapping to a physical register in physical register file 212. As noted above, more than one physical register may be mapped to a logical register. This number may be dependent upon the number of non-retired instructions. Some physicals registers in physical register file 212 may be dedicated for integers whereas others may be dedicated for floating point numbers, but for simplicity, these distinctions are not made.

During renaming of an instruction, current entries in RAT 210 may provide the required mapping, or mappings, for renaming the source logical register, or registers, of the instruction, and a new mapping is created for the destination logical register of the instruction. The new mapping may evict the old mapping in RAT 210, and RAT 210 is updated with this new mapping.

Renamed instructions are placed in instruction window buffer 226. Entries in instruction window buffer 226 contain the opcode of the renamed instruction, as well as other fields. The results of the instructions, however, may not be stored in instruction window 226 but may reside

in physical register file 212. Instruction window buffer 226 may allow for instruction retirement in original program order. Instruction window buffer 226 may operate as a circular buffer, where a first pointer points to the next instruction to be retired and a second pointer points to the next available entry for a newly decoded instruction. When an instruction retires, the first pointer is incremented. When a decoded instruction is placed in instruction window buffer 226, the second pointer is incremented. The pointers are incremented by one for each decoded instruction in a RISC architecture, and may be incremented by more than one for CISC architectures.

According to embodiments of the present invention, instruction window buffer 226 may have pointer fields for pointing to physical registers. When an instruction placed in instruction window buffer 226 has evicted a mapping in RAT 210, a pointer field associated with the instruction is set to point to the physical register of the evicted mapping. The pointer fields comprise an active list 216. Active list 216 includes old field 218 and new field 220. Free list 214 may be a queue of pointers pointing to unallocated physical registers. Free list 214 also may reclaim allocated registers from active list 216 as they become available.

Active list 216 may have two fields. Old field 218 stores the evicted registers from RAT 210. This process is similar to active list 102 depicted in Fig. 1. New field 220 stores, for each entry, the physical register newly allocated in RAT 210. Thus, active list 216 may record the allocated physical register from free list 214 for each instruction. Upon a misprediction or other exception within processor 200, free list 214 may be repaired by pushing back all physical registers pointed by new field 220 of all instructions past the mispredicted branch. Thus, in addition to active list 216 having evicted physical registers in old field 218, active list 216 also

has the newly allocated physical registers in new field 220. Therefore, free list 214 may reclaim registers from active list 216 without algorithms or indirect processes executed at a misprediction condition.

Fig. 4 illustrates a block diagram of a register renaming structure in accordance with an embodiment of the present invention.

Fig. 5 depicts a flowchart of a method for recovering registers in a processor in accordance with an embodiment of the present invention.

Fig. 6 depicts a flowchart of another method for recovering registers in a processor in accordance with another embodiment of the present intervention.

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Are claims 13, 14 and 16-18 anticipated under 35 U.S.C. 102(b) by Yeager et al. (U.S. Patent No. 5,758,112) ("Yeager")?

7. ARGUMENT

A. Claims 13, 14 and 16-18 are not anticipated by Yeager

First, Applicants would like to gratefully acknowledge Examiner's allowance of claims 1-5, 7-12 and 19-22, including independent claims 1, 8 and 19. *See* page 2, Office Action dated 7/18/2005.

Applicants respectfully submit the Yeager reference does not teach suggest or disclose "[a] method for recovering registers in a processor, comprising: reading a bit in an active list; *reclaiming a physical register from a new field of said active list to a free list* according to said

bit; and setting said bit during *a misprediction condition*" (e.g., as disclosed in the embodiment of claim 13).

In the original rejection of independent claim 13, the Examiner asserted that Yeager teaches a method for recovering registers in a processor (col. 2, lines 40-42); reading a bit in an active list (col. 15, line 61 to col. 16, line 14); reclaiming a physical register from said active list to a free list according to said bit (col. 16, lines 1-14). *See* page 3, Office Action dated 12/9/2003. Applicants disagree. Column 2, lines 40-42 state:

> The present invention offers a highly efficient mechanism for saving and restoring register-renaming information to facilitate branch prediction and speculative execution.

The cited section of Yeager does not disclose at least a reclaiming a physical register from a new field of said active list to a free list according to said bit; and setting said bit during a misprediction condition as specifically described in the embodiment of claim 1.

Column 15, lines 61-67 state:

> Done bit 286 and exception bit 288, as shown in FIG. 2, are initialized, usually to zero, when an instruction is decoded. These bits are set when an instruction completes execution. The "done" and "exception" bits control when instructions graduate. When "done," up to four instructions can graduate per cycle, provided no previous instruction had an exception.

The cited section discloses the "done" and "exception" bits of Yeager, but again does not disclose at least reclaiming a physical register from a new field of said active list to a free list according to said bit; and setting said bit during a misprediction condition, or any equivalent thereof, as specifically described in the embodiment of claim 13.

Finally, Column 16 lines 1-14 state:

> Instructions graduate in original program order from active list 212, after they have been completed by a functional unit. Instructions complete (i.e., become eligible for graduation) when their results have been computed and stored in a register. Because execution and completion can occur out of order, each completion is recorded by setting

> done bit 286 for that instruction. Address queue 308 generates "done" signals for load and store instructions. For all other instructions, an execution unit sets the done bit of an instruction in active list 212. As instructions graduate, a "graduation unit" (not shown) removes instructions from active list 212 and appends their old destination registers to free lists 208 or 210 for re-use. Up to four instructions can graduate in parallel during each cycle.

The cited section discloses the utility and operation of the "done bit" and a "graduation unit" of Yeager in the completion of instructions, but fails to disclose at least reclaiming a physical register from a new field of said active list to a free list according to said bit; and setting said bit during a misprediction condition or any equivalent thereof. Therefore, Applicants submit that none of the cited section in the previous Office Actions disclose a "misprediction condition" as described in embodiments of the present invention.

In the Office Action dated 11/24/2005, the Examiner allowed independent claims 1 and 19 (and the corresponding depending claims). After Applicants' responses dated 4/12/2005 and 5/12/2005, the Examiner allowed independent claim 8 (and the corresponding dependent claims) as well. *See* Office Action dated 7/18/2005.

Applicants submit that above and beyond the fact that Yeager reference does not disclose the limitations of independent claim 13, Applicants submit independent claim 13 contains similar allowable limitations to that found in previously allowed independent claims 1, 8, and 19, and should also be allowed for similar reasons. For example, claim 1 recites the limitation "[a] processor, comprising:...a misprediction condition wherein said free list is to reclaim mispredicted allocated said physical registers from said new field". Also, Moreover, amended claim 8 recites: "[a] method for recovering registers in a processor, comprising:... reclaiming during a misprediction condition allocated physical registers in said new field to a free list".

Similarly, claim 13 recites "[a] method for recovering registers in a processor, comprising:... reclaiming a physical register from a new field of said active list to a free list according to said bit; and setting said bit during a misprediction condition. Therefore, similar to claim 1, the embodiment of claim 13 describes (among other things) reclaiming a physical register from a new field of said active list to a free list according to said bit during a misprediction condition.

Nevertheless, in the arguments in the Advisory Action, the Examiner focuses singularly on the "misprediction condition" limitation of independent claim 13, instead of the totality of the allowable limitations found in the embodiment. In the Advisory Action, the Examiner specifically quotes column 17, lines 26-28 and the previously uncited column 1, lines 23-28 of Yeager as disclosing a "misprediction condition". Applicants disagree with the Examiner's assertion. However, assuming *arguendo* that this is case, it still would not be sufficient to support a proper §102(b) rejection, as simply disclosing a generic notion of a "misprediction condition" (only *arguendo*) is insufficient without disclosing the associated limitations.

In particular, Applicants submit that the cited sections do not teach *reclaiming a physical register from a new field of said active list to a free list* according to said bit; and setting said bit *during a misprediction condition* (limitations similar to those found in the allowable independent claims 1, 8 and 19). Column 17, lines 26-28 of Yeager state:

> As discussed above, redundant mapping tables 204' and 206' (FIGS. 3-7) make a copy of mapping tables 204 and 206, respectively, for use if a branch is restored.

Column 1, lines 23-28 state:

> This invention relates in general to computers capable of performing register renaming and branch prediction and, in particular, to a computer capable of restoring register-renaming hardware to a pre-branch state in a single clock cycle in the event of a misprediction.

The Examiner cites these two sections to assert that Yeager teaches "restoring" due to a "misprediction condition". Applicants respectfully disagree and submit that the generic use of the term "misprediction" is insufficient to disclose a "misprediction condition" as described in embodiments of the present application. Furthermore, regardless of whether Yeager teaches restoring conditioned upon a "misprediction condition", Yeager does not teach *reclaiming a physical register from a new field of said active list to a free list ... during a misprediction condition* as specifically described in *both* independent claim 13 *and* presently allowed independent claim 1. Therefore, Applicants submit that the Examiner has already allowed claims 1, 8, and 19 over the Yeager reference, and claim 13 should be allowed for similar reasons. An allowance should not be withheld simply because independent claim 13 allegedly recites "extremely minimal claim language" (*See* Office Action dated 7/18/2005, page 2, paragraph 4, line 5) when at least all of the relevant allowable limitations are specifically recited in the claim.

Appellants therefore respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's decision rejecting claims 13, 14 and 16-18 and direct the Examiner to pass the case to issue.

Serial No. 09/750,095
Appeal Brief Filed February 18, 2006
Advisory Action dated December 19, 2005

The Examiner is hereby authorized to charge the appeal brief fee of **$500.00** and any additional fees which may be necessary for consideration of this paper to Kenyon & Kenyon Deposit Account No. **11-0600**.

Respectfully submitted,

KENYON & KENYON LLP

Date: February 21, 2006 By: _____
Sumit Bhattacharya
(Reg. No. 51,469)

KENYON & KENYON LLP
333 West San Carlos St., Suite 600
San Jose, CA 95110
Telephone: (408) 975-7500
Facsimile: (408) 975-7501

APPENDIX

(Brief of Appellant Stephan J. Jourdan et al.
U.S. Patent Application Serial No. 09/750,095)

8. CLAIMS ON APPEAL

1. (Previously presented) A processor, comprising:

a physical register file populated by a number of registers;

an instruction decoder;

a register alias table coupled to the instruction decoder;

an active list populated by a number of entries, the entries include an old field and a new field;

a free list of unallocated physical registers reclaimed from said active list; and

a misprediction condition wherein said free list reclaims mispredicted allocated said physical registers from said new field.

2. (Previously Presented) The processor of claim 1, further comprising an instruction window buffer to store dispatched instructions.

3. (Previously Presented) The processor of claim 2, wherein said dispatched instructions correlate to evicted allocated physical registers, said free list is to reclaim said evicted physical registers when said dispatched instructions retire.

4. (Previously Presented) The processor of claim 1, wherein said instruction is to write to said allocated physical register.

5. (Previously Presented) The processor of claim 1, wherein said allocated physical register is to be allocated from said free list.

6. (Cancelled)

7. (Original) The processor of claims 1, further comprising a bit field within said active list, said bit field comprising at least one bit to indicate whether the instruction is retired correctly.

8. (Previously Presented) A method for recovering registers in a processor, comprising:

detecting an exception correlating to an instruction associated with an entry on an active list;

moving a pointer on said active list to an old field and a new field after said entry; and

reclaiming during a misprediction condition allocated physical registers in said new field to a free list.

9. (Previously Presented) The method of claim 8, further comprising flushing instructions in an instruction window buffer after said instruction associated with a misprediction condition.

10. (Original) The method of claim 9, further comprising overwriting entries in said active list.

11. (Original) The method of claim 8, further comprising allocating unallocated physical registers from said free list to a register alias table.

12. (Original) The method of claim 11, further comprising moving evicted physical registers from said register alias table to said active list.

13. (Previously Presented) A method for recovering registers in a processor, comprising:

reading a bit in an active list;

reclaiming a physical register from a new field of said active list to a free list according to said bit; and

setting said bit during a misprediction condition.

14. (Original) The method of claim 13, further comprising overwriting an entry in said active list.

15. (Cancelled)

16. (Original) The method of claim 13, wherein said reclaiming includes reading said physical register from an old field in said active list.

17. (Original) The method of claim 13, wherein said reclaiming includes reading said physical register from a new field in said active list.

18. (Original) The method of claim 13, wherein said reading includes reading said bit in a bit field within said active list.

19. (Previously Presented) A register renaming apparatus within a processor, comprising:

a register alias table;

a first set of registers to be renamed by said register alias table;

an active list having an old field and a new field that correlate to said registers; and

a free list comprising a second set of registers reclaimed from said active list; and

a misprediction condition wherein said free list is to reclaim mispredicted allocated said registers from said new field.

20. (Previously Presented) The apparatus of claim 19, wherein said first set of registers correlate to non-retired instructions.

21. (Original) The apparatus of claim 19, wherein said active list includes a bit field.

22. (Original) The apparatus of claim 19, further comprising a pointer for said active list.

9. EVIDENCE APPENDIX

No further evidence has been submitted with this Appeal Brief.

10. RELATED PROCEEDINGS APPENDIX

Per Section 2 above, there are no related proceedings to the present Appeal.